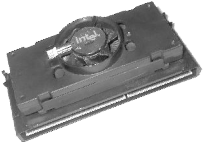


Tietokoneen rakenne

Luento 8

Suorittimen rakenne ja toiminta



Ch 12.1-4 [Sta06]

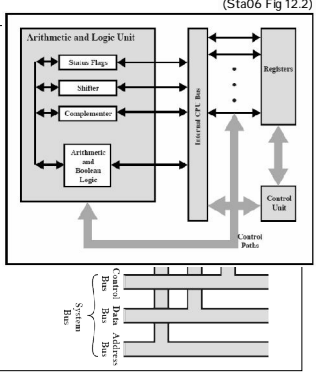
- n Rekisterit
- n Kääskysykli
- n Liukuhinna
- n Riippuvuusongelmat
- n Hyppyjen käsittely

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 1

CPU:n yleisrakenne

(Sta06 Fig 12.2)

- n ALU
 - u Laskenta, vertailut
- n Rekisterit
 - u Nopeaa työtalaa
- n Prosessoriväylä
 - u Bittien siirto paikasta toiseen
- n Ohjausyksikkö (Ch 16-17)
 - u Kuka? Mitä? Milloin?
 - u Kellopulssi
 - u Generoi ohjaussignaaleita
 - § mitä seuraavalla kellopulssilla tapahtuu?
- n MMU?
- n Välimuisti?



Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 2

Rekisterit

- n Muistihierarkian hupulla
- n Käyttäjälle näkyvät rekisterit
 - u Ohjelmoija/kaantaja päättää niiden käytöstä
 - u Paljonko? Nimeäminen? ADD R1,R2,R3
- n Ohjaus ja statusrekisterit
 - u Osa epäsuorasti ohjelman viitatavissa
 - § PC, PSW, flags, ... BNEQ Loop
 - u Osa vain CPU:n sisäiseen käyttöön
 - § MAR, MBR, ...
- n Sisäisiä apurekistereitä käskyn välivalheita varten
 - u Esim. - Kääskyrekkisteri (IR) käskyn tulkintaa varten
 - operandi ensin käskystä apurekkisteriin, vasta sitten ALU:uun

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 3

Käyttäjälle näkyvät rekisterit

- n Eri prosessoriperhe \emptyset eri määrä erilaisia rekistereitä, erilaiset nimeämistavat, erilaiset käyttötarkoitukset
- n Yleisrekisterit
- n Datarekisterit
- n Osoiterekisterit
- n Segmenttirekisterit
- n Pino-osoitin, ympäristöosoitin (frame pointer)
- n Status- ja tilarekisterit

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 4

Esimerkkejä

(Sta06 Fig 12.3)

Data Registers

D0
D1
D2
D3
D4
D5
D6
D7

Address Registers

A0
A1
A2
A3
A4
A5
A6
A7

Program Status

Program Counter
Status Register

(a) M68000

General Registers

AX	Accumulator
BX	Base
CX	Count
DX	Data

Pointer & Index

SP	Stack Pointer
BP	Base Pointer
SI	Source Index
DI	Dest Index

Segment

CS	Code
DS	Data
SS	Stack
ES	Extra

Program Status

Instr Ptr
Flags

(b) 8086

General Registers

EAX	AX
EBX	BX
ECX	CX
EDX	DX

General Registers

ESP	SP
EBP	BP
ESI	SI
EDI	DI

Program Status

FLAGS Register
Instruction Pointer

(c) 80386 - Pentium 4

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 5

PSW - Program Status Word

- n Nimi vaihtelee eri arkkitehtuureissa
- n CPU:n tila
 - u Etuoikeutettu vs. käyttäjätila
- n Vertailun tulos
 - u Esim. Greater, Equal, Less, Zero, ...
- n Sattuiiko käskyn suorituksessa poikkeuksia?
 - u Nollalajako, ylivuoto
 - u Page fault, "memory violation"
- n Tarvitseeko ulkoinen laite ohjausta?
 - u Bitti keskeytyksen ilmaisemiseksi
- n Keskeytyksen esto / salliminen
 - u Kullekin luokalle omat bittinsä

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 6

Käskysykli

(Sta06 Fig 12.5)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 7

Käskyn nouto (10)

- MAR • PC
- MAR • MMU(MAR)
- Control Bus • Reserve
- Control Bus • Read
- PC • ALU(PC+1)
- MBR • MEM[MAR]
- Control Bus • Release
- IR • MBR

MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

(Sta06 Fig 12.6)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 8

Operandin nouto, Epäsuora osoitus (13)

- MAR • Address
- MAR • MMU(MAR)
- Control Bus • Reserve
- Control Bus • Read
- MBR • MEM[MAR]

- MAR • MBR
- MAR • MMU(MAR)
- Control Bus • Read
- MBR • MEM[MAR]
- Control Bus • Release
- ALU? Regs? • MBR

Väliluisti!

(Sta06 Fig 12.7)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 9

Keskeytykskäsittelyn alkutoimet (esim.) (16)

- MAR • SP
- MAR • MMU(MAR)
- Control Bus • Reserve
- MBR • PC
- Control Bus • Write
- MAR • SP • ALU(SP+1)
- MAR • MMU(MAR)
- MBR • PSW
- Control Bus • Write
- SP • ALU(SP+1)
- PSW • privileged & disable
- MAR • Interrupt nbr
- Control Bus • Read
- PC • MBR • MEM[MAR]
- Control Bus • Release

SP = Stack Pointer (Sta06 Fig 12.8)

Ei osoitteenmuunnosta!

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 10

Tietokoneen rakenne

Liukuhihna

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 11

Esim. Pesula (by David A. Patterson)

- Ann, Brian, Cathy, Dave: kullakin satsi pyykättävää **A B C D**
- Pesu vie 30 min
- Kuivaus vie 40 min
- Viimeistely vie 20 min

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 12

Peräkkäin: Pyykkitupa varattu vain yhdelle (8)

n Vie 6 tuntia, syntyy neljä satsia puhdasta
 n Entäpä, jos toimisivat liukuhihnaperiaatteella?

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 13

Liukuhihna: Varaus työvaiheittain (13)

n Vie 3,5 tuntia, syntyy 4 satsia puhdasta

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 14

Huomioita

n Liukuhihna ei nopeuta yhden työn aktiivisena oloaikaa järjestelmässä
 u Mutta keskimääräinen läpimenoaika paranee
 n Liukuhihna voi aiheuttaa yhdelle työlle **läävitystä** verrattuna siihen, jos se työ olisi yksin järjestelmässä
 u Jos hihnan seuraava vaihe varattuna, ei voi edetä
 n Useita töitä etenee yhtä aikaa, mutta ne ovat eri vaiheissa
 n Hitain vaihe määrää koko liukuhihnan nopeuden
 u Liukuhihna nytkähtää, kun kaikki osat valmiita
 u Tehokkuus kärsii, jos kestot eripituisia
 n Yksi työ nopeutuu **maksimissaan saman verran kuin liukuhihnassa vaiheita**

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 15

Huomioita

n Toteutus monimutkaisempaa, **lisäresurssien tarve**
 u Tuplapistorasia? Ž pesukone ja kuivaaja voivat porrata yhtäaikaan
 u Kaksi tai kolme ihmistä yhtäaikaan pyykituvassa
 n **Hihna ei toimi alussa eikä lopussa täydellä teholla**
 n **Hihna "yskii"**
 u Töitä ei tule tasaisella tahdilla

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 16

CPU ja 2-vaiheinen liukuhihna?

(Sta06 Fig 12.9)

n Seuravan käskyn **ennaltanouto samaan aikaan** kuin edellistä suoritetaan
 u Paikallisuus: oletetaan, että suoritus "esiintymisjärjestyksessä"
 n **Ongelmat**
 u Suoritusvaihe voi kestää kauemmin Ž noutovaihe joutuu
 u Suoritusvaihe muuttaa PC:n arvoa Ž haettu väärä käsky
 § Seuraavan käskyn osoite "ennustettiin" väärin!
 n **Rinnakkaisuus vielä vähäistä Ž Lisää vaiheita?**

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 17

6-vaiheinen liukuhihna

Instruction	Time													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

FE - Fetch instruction
 DI - Decode instruction
 CO - Calculate operand addresses
 FO - Fetch operands
 EI - Execute instruction
 WO - Write operand

(Sta06 Fig 12.10)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 18

Paljonko liukuhinna nopeuttaa?

- 6 vaiheinen liukuhinna ≈ 14 aikayksikköä
- Ei liukuhinnaa $\approx 9 \cdot 6 = 54$ aikayksikköä
- Nopeutus = $54/14 = 3.86 < 6!$
- Todellisuudessa nopeutus vieläkin pienempi
- Jokainen käsky ei tarvitse kaikkia vaiheita
 - Ei-liukuhinnoitettu voi edetä heti seuraavaan vaiheeseen
 - Liukuhinnoitettu joutuu odottamaan seuraavan vaiheen vapautumista, ko. CPU:n yksikkö jouten
 - Suorituksessa on "kupla" (bubble)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 19

Yhden vaiheen kesto liukuhinalla?

$$\tau = \max_{i=1..k} [\tau_i] + d = \tau_m + d \gg d$$

Yhden vaiheen kesto Vaiheen i kesto Viive vaiheesta toiseen siirtymiselle ~ yksi kellopulssi Hitaimman vaiheen (max) kesto

- Hitain vaihe määrää etenemisvauhdin
 - Kauimmin kestävä vaihe on pullonkaula
- Kukin vaihe asetetaan kestämään yhtä kauan
 - Montako kellopulssia?

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 20

Paljonko nopeuttaa?

n käskyä, **k** vaihetta, τ =vaiheen kesto

Ei liukuhinna: $T_1 = nk\tau$ Pessimistinen: olettaa, että kaikki vaiheet kestävät yhtä kauan

Liukuhinna: $T_k = [k + (n-1)]\tau$ ks. Sta06 Fig 12.10 ja tarkista itsesi

k vaihetta ennenkuin ensimmäinen työ valmistuu seuraavat (n-1) valmistuvat yhden vaiheen välein

Nopeutus: $S_k = \frac{T_1}{T_k} = \frac{nk\tau}{[k + (n-1)]\tau} = \frac{nk}{[k + (n-1)]}$

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 21

Paljonko nopeuttaa?

Oletuksella: ei hyppykäskyjä

(Sta06 Fig 12.14)

more gains from multiple stages when more instructions without jumps

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 22

Muita huomioita

- Lisävaiheita**
 - CPU:n laitettava välitulos jonnekkin
 - Vaikka yksittäinen käsky etenisi yksin hinnalla, sillä kuluu enemmän aikaa kuin ei-liukuhinnoitetussa
- Silti**
 - Ison käskyjoukon suorittaminen sujuu nopeammin
 - Parempi läpimenoaste (käskyä/sec)
- Liukuhinnoituksen rinnakkaisuus nopeuttaa kokonaisuutta, mutta hidastaa yhden käskyn suoritusta**

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 23

Ongelmakohtia

- Rakenteellinen riippuvuus** (structural dependency)
 - Usea vaihe voi tarvita samaa HW-yksikköä
 - Muisti: F1, FO, WO
 - ALU: CO, EI
- Kontrolliriippuvuus** (control dependency)
 - Ehdollisen hyppykäskyn tulos selvillä vasta EI-vaiheen jälkeen
 - \approx Ennaltanoudettu väärä käskyjä
- Datariippuvuus** (data dependency)
 - Tarvitaan edeltävän käskyn tulosta, eikä se ole vielä valmistunut

```

STORE R1,VarX
ADD R2,R3,VarY
MUL R3,R4,R5

ADD R1,R7,R9
Jump There
ADD R2,R3,R4
MUL R1,R4,R5

MUL R1,R2,R3
LOAD R6, Arr(R1)
    
```

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 24

Ratkaisuja?

- Huomattava laitteistolla, odotettava, että riippuvuus poistettu**
 - Hihnalle ylimääräisiä odotusjaksoja eli kuplia, "bubbles"
 - Kupla viivästää kaikkia sen takana tulevia vaiheita
 - Yleisesti käytetty tapa
- Rakenteellinen riippuvuus**
 - Lisää laitteistoa, esim. CO- ja EI-vaiheelle omat ALUt
 - Paljon rekistereitä, vähemmän muistioperandeja
- Kontrolliriippuvuus**
 - Tyhjennä hihna, ja hae uudet käskyt
 - Hyppyjen ennustuslogiikka: haenko ennalta tästä vai tuolta?
- Datariippuvuus**
 - Vaihda käskyjen suoritusjärjestystä
 - Oikopolut laitteistossa (by-pass): tulokseen pääsee käsiksi ennen WO-vaihetta

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 25

Datariippuvuudet ja niiden poisto

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 26

Käskyjen suoritusjärjestyksen muutos

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 27

Oikopolut

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 28

Tietokoneen rakenne

Hypyt ja liukuhihna

- Viivästetty hypy
- Monta suorituspolkua
- Kohteen ennaltanouto
- Silmukkapuskuri
- Ennustuslogiikka

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 29

Hypyn vaikutus liukuhihnaan

(Sta06 Fig 12.11) (Sta06 Fig 12.12)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 30

Viivästetty haarautuminen (delayed branch)

- n Kääntäjä laittaa hyppykäskyn perään muita käskyjä, jotka suoritetaan aina
 - u Hihnalle ei päästetä mitään, mikä pitäisi ehkä myöhemmin perua
 - § tehdyn työ peruuttaminen on tosi vaikeata!
 - u Jos ei hyödyllisiä käskyjä, sitten NOP-käskyjä
- n Jos hyppy toteutuu, nuo käskyt jo hyvässä vauhdissa ja annetaan niiden valmistua
 - u Hihnaa ei tarvitse erikseen tyhjentää
- n Jos hyppy ei toteudu, NOP-käskyt vain hukkaavat syklejä
- n Helpompi toteuttaa näin kuin tyhjentää hihna

```

sub r5, r3, r7
add r1, r2, r3
jump There
...
↓
sub r5, r3, r7
jump There
add r1, r2, r3
...

```

delay slot

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 31

Monta suorituspolkua (multiple instr. streams)

- n **Spekuloi**
 - u Hae hihnalle hyppykäskyä seuraavia käskyjä
 - u Hae toiselle hihnalle käskyjä hypyn kohdeosoitteesta
- n **Anna polkujen edetä kunnes selvää mikä oikea**
 - u Hylkää väärä polku (tai ainakin sen tulokset)
- n **Ongelma**
 - u Hypyn kohde saattaa selvitä vasta osoitelaskennan jälkeen
 - u Polku voi haarautua edelleen
 - § Jatketaanko samoin? Yksi spekulointi kerrallaan?
 - u Tarvitaan lisälaitteistoa
 - § Useampia liukuhinoja
 - § Spekulatiivisia tuloksia ei saa tallettaa oikeisiin rekistereihin
 - u Spekulointi käyttää resursseja ja voi hidastaa oikeaa työtä
 - § CPU:n sisäiset työrekiesterit, väylä, ALU
- n **Liukuhinna osattava tyhjentää**

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 32

Kohteen ennaltanouto (prefetch branch target)

- n Nouda etukäteen käsky hypyn kohteesta, mutta älä päästä sitä muuten liukuhihnalle
 - u Tee siis vain FI-vaihe
 - u Jos hyppy toteutuu, säästyy muistintuotoon muuten tarvittava aika
- n Liukuhinna osattava tyhjentää IBM 360/91 (1967)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 33

Silmukkapuskuri (loop buffer)

- n Pidä n viimeisintä käskyä CPU:n sisällä nopeassa puskurissa
 - u Hyödynnä myös ennaltanoutoa
 - § Hyvällä tuurilla myös hypyn kohdekäsky tuli samalla
 - § Esim. IF-THEN ja IF-THEN-ELSE rakenne
- n Toimii hyvin pienille silmuille
 - u Nouto muistista vain kerran
- n Parempi alueellinen paikallisuus kuin pelkkää välimuistia käytettäessä

CRAY-1
Motorola 68010

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 34

Hyppyjen ennustus (branch prediction)

- n Arvaa älykkäästi kumpi todennäköisempää: hyppy vai ei hyppää?
- n **Staattinen ennustus**
 - u Kiinteästi: aina hypätään
 - u Kiinteästi: ei koskaan hypätä **Motorola 68020**
VAX 11/780
 - § ~ 50% oikein
 - u Operaatiokoodin perusteella
 - § Selvitetty etukäteen (ennen CPU:n valmistusta) operaatiokoodit, joilla hyppy yleisempää kuin ei hyppää
 - § Esim. BLE käskyä käytetään tyypillisesti "steppailevan" for-silmukan lopussa, joten arvaa että hyppää
 - § ~ 75% ennusteista oikein

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 35

Hyppyjen ennustus

- n **Dynaaminen ennustus**
 - u Muistele miten tämän suorituskerran aikana kävi kyseisen hyppykäskyn kanssa aiemmin
 - § Ennustus osuu nyt paremmin kohdalleen
 - u Tarvitsee CPU:n sisälle apumuistia = branch history table
 - § Käskyn osoite
 - § Hypyn kohdeosoite (tai kohdekäsky)
 - § Hypätäänkö vai ei: taken / not taken
- n **"Karvalakkimalli"**
 - u Ennusta sen mukaan kuinka kävi edellisellä kerralla
 - § 1 bitti riittää
 - u Silmukoissa tulee 1 tai 2 väärää ennustusta

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 36

Hyppyjen ennustus

Parannettu malli

- Älä muuta ennustetta liian helposti
- Ennusta kahden edellisen kerran mukaan
- 2 bittia riittää

PowerPC 620

```

    graph TD
      T1((Predict Taken)) -- Taken --> T1
      T1 -- Not Taken --> TN1((Predict Not Taken))
      TN1 -- Taken --> T1
      TN1 -- Not Taken --> TN1
  
```

(Sta06 Fig 12.17)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 37

Hyppyjen ennustus

Which branch instruction is this?

IPFAR = instruction prefix address register

Prediction: taken/not taken

Where to jump, if branch taken

(Sta06 Fig 12.18)

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 38

Kertauskysymyksiä

- Mitä tietoja on sisällytettävä PSW:hen?
- Miksi 2-vaiheisesta liukuhihnasta ei ole paljon hyötyä?
- Mitkä tekijät vaikeuttavat liukuhihnan toimintaa?
- Millaisia ratkaisuja on käytetty hyppykäskyjen vaikutuksen eliminoimiseen?
- Kuinka CPU siirtyy keskeytyskäsitteeseen?

Tietokoneen rakenne / 2006 / Teemu Kerola 26.9.2006 Luento 8 - 39